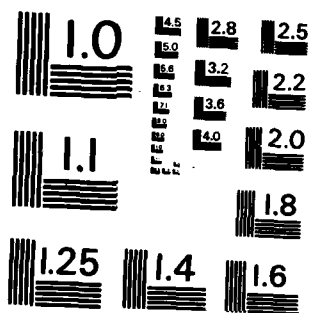


AD-A122 438 INVESTIGATION OF THE ADA LANGUAGE IMPLEMENTATION OF THE 1/1
HELLENIC COMMAND CONTROL AND INFORMATION SYSTEM(U)
UNCLASSIFIED JUN 82 NAVAL POSTGRADUATE SCHOOL MONTEREY CA A KOUTSOTOLIS
P/G 9/2 NL

END
DATE
FILMED
1 83
DT 6



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

2

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

INVESTIGATION OF THE ADA LANGUAGE
IMPLEMENTATION OF THE HELLENIC
COMMAND CONTROL AND INFORMATION SYSTEM

by

Apostolos Koutsotolis

June 1982

Thesis Advisor:

U. R. Kodres

Approved for public release; distribution unlimited.

STATIC
ELECTE
DEC 14 1982
A

ADA 122 435

FILE COPY

82 12 16 025

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. A122430	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Investigation of the Ada Language Implementation of the Hellenic Command Control and Information System		5. TYPE OF REPORT & PERIOD COVERED Master's thesis; June 1982
7. AUTHOR(s) Apostolos Koutsotolis		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE June 1982
		13. NUMBER OF PAGES 77
		15. SECURITY CLASS. (of this report) Unclassified
		16a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release. distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Ada Standardization Program Application of Ada Command Support Concept to the Command Control Command Control and Information System and Information System		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) → This thesis examines the features of the Ada language, describes the structure of the Hellenic Command Control and Information System (HCCIS) and investigates the use of Ada for the program development of HCCIS. The Ada high order programming language system is being procured to act as a standard for the implementation of →		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6001

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(20. ABSTRACT Continued)

future United States embedded computer systems. Many benefits are claimed from this approach for software engineering and management practice. HCCIS is a future system which will provide a network of automatic data processing support at Commands. ←



Accession Date	
Serial Number	
DTIC Number	
Classification	
Availability	
Abstracted	
Availability of the	
Abstract	
DTIC	Special

A

Approved for public release; distribution unlimited.

Investigation of the Ada Language
Implementation of the Hellenic
Command Control and Information System

by

Apostolos Koutsotolis
Commander, Hellenic Navy

Submitted in partial fulfillment of the
requirements for the degree of

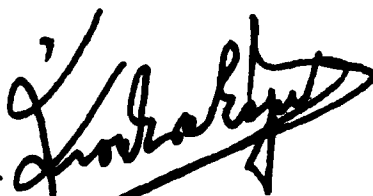
MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

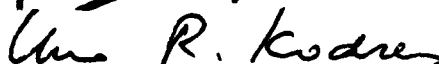
NAVAL POSTGRADUATE SCHOOL

June 1982

Author:



Approved by:



Thesis Advisor



Second Reader



Chairman, Department of Computer Science

W. M. Woods
Dean of Information and Policy Sciences

ABSTRACT

This thesis examines the features of the Ada language, describes the structure of the Hellenic Command Control and Information System (HCCIS) and investigates the use of Ada for the program development of HCCIS.

The Ada high order programming language system is being procured to act as a standard for the implementation of future United States embedded computer systems. Many benefits are claimed from this approach for software engineering and management practice. HCCIS is a future system which will provide a network of automatic data processing support at Commands.

TABLE OF CONTENTS

I.	INTRODUCTION -----	9
A.	PROBLEMS OF SOFTWARE ACQUISITION -----	9
B.	PROJECT MANAGER'S PERSPECTIVE -----	10
C.	HCCIS AS AN APPLICATION -----	12
II.	THE ADA STANDARDIZATION PROGRAM -----	14
A.	THE PROGRAM FOR ADA AVAILABILITY -----	14
B.	A DESCRIPTION OF THE ADA LANGUAGE SYSTEM ----	16
1.	The Program Developed Concept -----	16
2.	The Features and Facilities Provided by Ada -----	17
3.	The Facilities Provided by the APSE ----	21
4.	Portability Issues -----	25
5.	The Use of Ada and Its Support Tools ----	27
C.	SOFTWARE ENGINEERING IMPLICATIONS -----	29
1.	Benefits Using Ada -----	29
2.	Penalties Using Ada -----	30
D.	ADA'S CONTRIBUTION TO SOFTWARE QUALITY AND DEVELOPMENT PRODUCTIVITY -----	31
1.	Software Quality -----	31
2.	The Impact of Ada on Software Quality ---	34
3.	Development Productivity -----	34
4.	Definitions of Software Quality Attributes -----	36
III.	THE HELLENIC COMMAND CONTROL AND INFORMATION SYSTEM (HCCIS) -----	42
A.	A DEFINITION OF HCCIS -----	42

B. COMMAND SUPPORT CONCEPT -----	43
1. Receipt of Mission -----	45
2. Commander's Analysis and Mission -----	45
3. Collection of Information -----	45
4. Preparation of Planning Guidance -----	45
5. Development of Courses of Action -----	46
6. Development of Staff Estimates -----	46
7. Briefing the Commander -----	46
8. Commander's Decision -----	47
9. Preparation of the Concept of Operations --	47
C. HARDWARE CONFIGURATION -----	47
1. Microcomputers -----	48
2. Dynamic Display Terminals and Large Screen Display -----	48
3. Printers -----	49
4. Communications Control Panel -----	49
D. SOFTWARE CONFIGURATION -----	50
1. Communications -----	50
2. Data Distribution -----	52
3. Messages and Reports -----	55
4. Information Processing Tools -----	55
5. Tactical Processing Tools -----	56
6. Continuity of Operations -----	57
E. PERSONNEL -----	59
F. RELIABILITY, MAINTAINABILITY -----	60
1. Reliability -----	60
2. Maintainability -----	61

G.	SECURITY -----	61
1.	Physical Protection -----	61
2.	Error Detection -----	62
3.	Operational Features -----	62
4.	Access Control -----	62
H.	SOFTWARE ENGINEERING REQUIREMENTS -----	62
IV.	APPLICATION OF ADA/APSE TO THE HCCIS -----	64
A.	TECHNICAL COMPARISON -----	64
1.	The Programming of Each Computer Type to be Used -----	64
2.	Large Programs Development -----	65
3.	"Off-the-Shelf" Hardware-Independent Software Packages -----	66
4.	High Quality Software -----	66
5.	Real-Time Processing -----	68
6.	Security of Data -----	68
7.	Concurrent Execution of the Same Program -----	69
8.	Structured Programming -----	69
B.	ALTERNATIVES TO ADA -----	70
1.	Assembly -----	70
2.	FORTRAN -----	70
3.	CMS-2 -----	71
4.	JOVIAL J73 -----	71
V.	CONCLUSIONS -----	73
	LIST OF REFERENCES -----	75
	INITIAL DISTRIBUTION LIST -----	77

ACKNOWLEDGMENT

I would like to thank Professor Uno Kodres. Without his patience and guidance this thesis would never have come to be.

I. INTRODUCTION

The objective of this thesis is to examine the Ada programming language system and how well it can meet the software requirements for the implementation of the Hellenic Command Control and Information System (HCCIS).

A. PROBLEMS OF SOFTWARE ACQUISITION

The procurement of software for military computer systems has proved over the years to be a difficult task. Studies of military computer systems tend to show that not only is the cost of the software disproportionately high in comparison with the other system components, but that the software also accounts for a significant number of cases of poor performance or failure of the system at some stage during its lifecycle.

Also, a military organization probably will experience several of the following shortcomings in respect to the computer systems that it procures:

- (1) There is cost overrun and lateness of products.
- (2) The system is expensive to operate and maintain.
- (3) The system is inflexible with respect to enhancements and evolution.
- (4) There tends to be a mis-match between the system functions and the operational requirements.
- (5) There are latent faults in the software.

In addition to their financial implications, such shortcomings have other serious consequences for the military

organization. The mis-match between function and requirement means that the system cannot do its best and its performance in reacting to abnormal conditions is poorer than it should be. The presence of latent faults could mean that the computer system cannot react to a situation at all, or that it might react incorrectly.

In summary, the operational effectiveness of the entire military organization is decreased because of computerized systems.

B. PROJECT MANAGER'S PERSPECTIVE

Software engineering is defined as the science of design, development, implementation, test, evaluation and maintenance of computer software over its life cycle. It is the set of principles and procedures which result in the development of optimally acceptable software that is reliable and efficient on real machines.

Recognizing how much software engineering will help him, a project manager will look for techniques which will achieve improved performance in the following areas:

- (1) Controlability: the chosen technique leads to better control over the work of the development team.
- (2) Productivity: the chosen technique will obtain improved productivity from the software team, for example with more automation or re-utilization of existing software.
- (3) Quality: the product will be of good quality and many serious problems will be avoided after delivery.

Thus it is easier for a project manager choosing the appropriate techniques to measure the progress of the software

development against well definable criteria and identify the advantages of this technique over the alternatives.

As one of the larger software procurers, the United States Department of Defense (US DoD) certainly experiences and appreciates the problems of software acquisition. In the mid-1970s, its expenditure in software was over three billion dollars annually, of which over 50% was spent in software for embedded computer systems (those applications in which the computer is integrated into an electronic or electro-mechanical system, e.g., a weapon system, from a design, procurement and operational viewpoint).

The DoD deemed that the cost effectiveness was poor, so it initiated a research and development program in order to improve the future procurement and life cycle cost performances. This program led to the decision for standardization on a single programming language for use in embedded systems with full support facilities for software development. The DoD soon established that no existing language system was suitable for standardization and therefore made the decision to procure a new package.

This effort has yielded the high order programming language Ada, the design of which was completed in 1980. The effort is now continuing with the final definition and development of the support facilities, the Ada Programming Support Environment (APSE) from which initial experimental components are now becoming available. The international

computing community is actively participating and monitoring the developments related to the Ada language and its support environment.

C. HCCIS AS AN APPLICATION

Hellenic forces as a software procurer, are interested in development the Hellenic Command Control and Information System (HCCIS). This is intended to provide computer assisted support to the command and control processes within several Commands. HCCIS supports all missions which may be assigned to Hellenic Forces under all operating conditions that could conceivably be encountered in any part of the Hellenic area. Using modular equipment, HCCIS has the flexibility to be configured to support any task organization regardless of size and composition. The system operates to support training exercises, operational planning and execution of combat operations. HCCIS supports Commanders and their staffs in carrying out functions in the areas of operations and intelligence, including planning, intelligence production, and the monitoring and directing of tactical operations. The system provides this support to ground elements, sea elements, air elements and headquarters.

The software engineering, software management and time-frame give Ada and APSE a clear potential relevance to the HCCIS implementation. It is the principal objective of this thesis to specify that relevance. This is achieved by first making a technical appraisal of both the Ada and HCCIS

programs, and then examining the requirements of HCCIS for compatibility with the facilities offered by Ada and APSE.

Chapter II is devoted to describing the major features and facilities of the Ada language and its support environment. In Chapter III the characteristics and requirements of the Hellenic Command Control and Information System for software support are examined and evaluated. Chapter IV compares the two programs for compatibility. Chapter V summarizes and concludes whether or not the Ada language can be used as the HCCIS programming language.

II. THE ADA STANDARDIZATION PROGRAM

In this chapter the goals of the United States Department of Defense's (US DoD's) Ada program are summarized, the development stages up to the present day are described, and future planning examined. From this, an estimation is made when the various products might become available. Thereafter, these products are examined and their value to software engineering and software management described to indicate their importance.

A. THE PROGRAM FOR ADA AVAILABILITY

The overall scope and aims of the program to develop a new programming language (Ada) can be stated as:

- (1) To define the requirements and the design specifications of a high order language for the complete programming of dedicated military systems software.
- (2) To define the requirements and the specifications of a program development support environment for Ada applications (i.e., the Ada Programming Support Environment (APSE)).
- (3) To ensure widespread acceptance of Ada and APSE and their use by the military services.
- (4) To procure the APSE in a form which enables easy portability across computer types.
- (5) To establish means of controlling changes to the Ada standard (i.e., through the Ada Language Control Board (ALCB)).
- (6) To provide a means for the validation of a new Ada compiler.
- (7) To provide instructional and training materials for Ada and APSE.

In July 1980, the design specification of the Ada language was finalized. To reach this point the specification went through several refining iterations. The requirements were defined in five stages of development, summarized in the Steelman document used to evolve the design.

The Ada language reference manual, published in 1980, represents the language specification, and the DoD has already initiated the procurement of tools for the Ada Compiler Validation Capability (ACVC) which consists of tests, tools, procedures, and documentation design to enforce and encourage development of compilers that conform to the Ada language standard.

The APSE specification is currently being evolved. A requirements document named Stoneman, published in 1980, specifies the structure and content of an APSE to support the development and maintenance phases of a system. As a result of its policy of encouraging international participation in all phases of the Ada program, there are implementations of APSE-like systems (including compilers) being sponsored by other country's computing communities. Also, many universities and private enterprises in the United States and Europe are developing compilers. The DoD has stated its intention to make Ada the only language authorized for their embedded systems from about 1990 onwards.

Experience gained from previous language standards shows that it takes about a decade for a language to achieve widespread use, and there is no indication to date that Ada

will achieve widespread use faster than any other high order language. By monitoring the progress of the whole international Ada program, we can attempt to predict the following future events:

- (1) Initial Ada compilers will be available from university, commercial and government sources starting from early 1983.
- (2) The Ada Compiler Validation Capability will be operational in 1983.
- (3) Experimental systems containing the essential basic facilities, called the Minimal Ada Programming Support Environment (MAPSE), will be available by 1984.
- (4) It can be expected that packages will become available to support particular application areas, such as database management, network handling etc., from about 1984 onwards, as has been scheduled by the Computer Corporation of America.

B. A DESCRIPTION OF THE ADA LANGUAGE SYSTEM

Paragraphs 1 to 5 combine to form a description of the features and facilities of the Ada language and its support system and how they might be used.

1. The Program Development Concept

Ada promotes the host/target concept of program development in which the Ada Programming Support Environment operates on a large scale computer system, the host. Ada programs are prepared, compiled, linked, tested (as far as possible) and maintained on the host. The target computer (the hardware on which the application will eventually run) need only be involved in the final stages of program and interface testing. For some projects, the target computer

might be of the same type as the host. This would be an advantage, but in most cases, the operational environment would be different from the program development environment (APSE).

2. The Features and Facilities Provided by Ada

The Ada language has been designed to provide all the features needed for complete real-time systems programming of the target computer. Therefore Ada programs do not need to rely on the target computer having additional software packages. In order to support this, the language features include:

a. Modern high level language for sequential coding of program modules, which includes the following features:

- (1) The syntax encourages a descriptive-English style of text.
- (2) Subtypes. A type characterizes a set of values that objects of the type may assume and a set of operations which can be applied to those values. It is possible to restrict the set of allowed values of a type without changing the set of applicable operations. Such a restriction is called a constraint and the subset of values it defines is called a subtype. Subtype is a subset of elements of a given type. Since subtype declarations do not introduce new types, objects of different subtypes of the same type are compatible for assignment. Subtypes are characterized by constraints which include range, precision, scale, index ranges, and user defined constraints.
- (3) Encapsulation (data abstraction) provides a means for limiting the accessibility of data items within a program. In the model where entities are either public (if declared in the visible part) or totally hidden (if declared in the module body), encapsulation

is referred to data types which correspond to a situation in which we want the name of a type to be public, but where the knowledge of its internal properties is to be available only to the module body. This encapsulation is achieved by declaring the type name within the visible part, but at the same time specifying the type to be private.

(4) Complete control structure mechanisms (conditional statements, loops) are provided.

(5) Procedures and functions are also provided.

b. Parallel tasking and inter-task communications, a large and complex area which organizes, through language constructs, the scheduling and the executive and communication features for controlling the run-time system. A task is a textually distinct program unit which may be executed concurrently with other tasks. It is similar in form to a package module. The major difference between a package module and a task module is that the package is merely a passive construct while a task is active. A task may be declared within any declarative part (except the visible part of another task) and similarly contains two distinct pieces of text. These are the specification part which describes its external appearance and the module body which describes its internal behavior.

c. Input/output, with high level facilities providing means for convenient transport of character and numeric data, and with low level facilities, especially in embedded computer systems, providing direct signal processing and interaction with non-standard peripheral devices.

d. Exception handling, which provides the mechanisms for the handling of unexpected program conditions, faults etc., and gives the capability for recovery. The ability to handle error situations is essential for reliability of real time systems. Exceptions provide a way to abnormally terminate a program unit. They are declared by an exception declaration of the form "NAME: exception". They are invoked by raising the exception by means of a statement "raise NAME" which is similar to a goto statement in that it transfers control out of the environment of invocation without any possibility of return. The action to which an exception gives rise is specified by an exception handler. The exception handler invoked by a raised exception is determined dynamically by propagating the exception backwards through the chain of calls until an exception handler which handles the raised exception is found. The exception handler serves as a terminator of the program unit in which it occurs, returning control to its caller on completion.

e. Generic definitions, which provide the ability to parameterize program units with the appropriate data types. Generic clauses provide a facility for translation time parameterization of program units. The primary purpose of this generic facility is factorization, so as to bring about a reduction in the size of the program text, while simultaneously improving both readability and efficiency. A generic instantiation creates an actual instance of the

specified program unit by replacement of the generic parameters. Such an instantiation must appear in a declaration which gives the name of the particular instantiation. For example the generic package PERSON, defines a template from which the instance JOHN can be created by the generic instantiation "package JOHN is new PERSON".

f. Declaration of target computer dependencies, which gives practicality to the language by enabling applications to map into particular hardware configurations.

g. Separate compilation (a program unit that is being compiled is able to get information from previous separately compiled units) and library facilities. The Ada library and subsequent language support programs provide the means for constructing executable Ada programs. The facilities provided may include:

- (1) The Ada library, unique to each target computer supported, including:
 - (a) The standard environment for the Ada language.
 - (b) The run-time system supporting the parallel processing and standard peripheral drivers.
 - (c) The packages for input/output.
- (2) Backing store overlays for code units (useful when target computers have no virtual memory hardware).
- (3) Interfacing to routines of other language systems.
- (4) Interfacing to other Ada packages (e.g., database management system).

From this list of features it can be appreciated that Ada is a large language, but this is due to its wide scope rather than a redundancy of ideas. It is the first language which includes all these features integrated into its design.

3. The Facilities Provided by the APSE

To fully realize the benefits of language commonality, a common programming environment is also required. This permits programmers to move from one host system to another, continuing to employ the same development tools and user interface. Concurrently with the Ada language design, the DoD conducted a program to determine the requirements for an integrated Ada environment, resulting in the Stoneman environment specification.

The APSE is to offer a well coordinate set of tools to support a programming project throughout its life cycle activity. It must be highly portable and employ conventions for interface between user and tool. Stoneman introduces the notion of a common open-ended database to serve as the interface through which a set of software tools can communicate.

For reasons of portability, Stoneman recognized three distinct levels within the environment: the Kernel Ada Programming Support Environment (KAPSE), the Minimal APSE (MAPSE), and the APSE.

The KAPSE, is designed to provide a machine independent interface to other APSE tools. All APSE tools using a common

KAPSE should prove portable for the set of environments supported by the KAPSE.

Stoneman defines also a minimum set of functions which an APSE must perform. This Minimal APSE (MAPSE) must provide a method to create database objects, modify database objects, transform an object from one representation to another, support the display objects, parse, link, load, and execute.

An APSE is a full environment based upon a particular MAPSE. However, it can incorporate additional tools of general interest, of interest only within a particular project, or of interest only to an individual programmer. In the case where no tools have been added, a MAPSE is itself an APSE. The APSE provides comprehensive, self-contained facilities for Ada program development and facilities that support the software management function.

a. Principal Characteristics of APSE

The principal characteristics and facilities expected to be offered to the user can be summarized as follows:

- (1) APSE runs on a powerful, large capacity host computer system.
- (2) APSE provides its users with a self-contained Ada oriented environment which operates under an existing host operating system. Once the user is in the APSE environment, the host system becomes invisible.
- (3) APSE provides time sharing facilities to on-line terminal users as well as remote entry to batch job running.

- (4) A powerful command language is provided to the user for communication with the APSE.
- (5) Database facilities are provided for the storage of any category of information. Information belonging exclusively to different projects can be kept apart.
- (6) Configuration control can be applied to categories of stored information (i.e., proper configuration control can automate several of the most time consuming and error prone activities of the operation and maintenance phase).
- (7) The Ada compiler front-end provides separate compilation of source modules into the standard intermediate code form. The associated Ada program library function organizes access and storage of Ada modules in source or intermediate form and controls version consistency.

b. Facilities for Target System

For each target system supported, the following facilities are provided:

- (1) Target run-time system library (as Para. B.2.g.(1)).
- (2) Target code generation from the intermediate representation and code optimization.
- (3) Link editing of the separately compiled modules into a single load unit, identifying backing-store overlays.
- (4) Loading of programs for execution.
- (5) Facilities for setting a loaded program into execution and interacting with it, via the connected peripherals.
- (6) An on-line debugging tool, whereby the executing program can be inspected and its operation controlled.
- (7) Debugging tool for symbolic execution, whereby the execution sequence and error states of a program may be related back to the Ada source text.

The facilities described up to this point are deemed to be the essential minimum for adequate software project support and therefore are called the Minimal Ada Programming Support Environment (MAPSE).

c. Additional Tools and Facilities for APSE

APSE, as mentioned above, is designed to be open-ended, so that tools and facilities can be added without difficulty. As more projects make use of APSE it is anticipated that tools will be developed to meet general requirements, as well as requirements more oriented to specific needs. Then, these would be made available to all APSE users. The kind of tools and facilities which might be expected in a mature APSE are:

- (1) Validation aids.
- (2) Branch testers.
- (3) Fault analysis aids.
- (4) Software development systems, in which all project information is held and its progress reported.
- (5) Requirement specification language system.
- (6) Document word-processing, with facilities to aid the documentation of Ada word-processing, facilities to aid the documentation of Ada programs, and military documentation structures.
- (7) Emulators for target computers. These are valuable tools for testing, because they can be used in addition to or instead of the actual target.

4. Portability Issues

Portability is a fundamental design criterion for Ada and APSE. It is the intention to make Ada and APSE transportable among a wide variety of large development systems so that development personnel do not need to be re-trained whenever systems are changed.

a. Ada Target Program Portability

The Ada language and its defined library packages provide all the major constructs necessary for structuring a real-time system program as a self-contained unit for execution on a target machine. Thus the program need not be reconstructed to move it from one target of conventional architecture to another, although local optimization might be desirable. This is the so-called retargetability, which refers to making the environment support a new target machine. It is important that the full language be properly supported for the new target.

Ada also relieves many of the traditional target to target incompatibility problems. Most of them are relieved by Ada's provision of library packages which support machine dependent physical interfaces, interrupt structures or special devices.

Thus, programs and libraries written in Ada should have better portability than other higher level languages.

b. The APSE Portability

Maximum portability is achieved by structuring the APSE into the Kernel APSE and the APSE toolset. The

KAPSE is largely invisible to the user. Its function is to provide a machine independent environment for the loading and execution of Ada programs on a host computer. The features that it provides are:

- (1) The interfacing functions with the host system, which may be a computer plus operating system or a bare machine without other software support.
- (2) A host independent multi-programming environment for Ada program development, loading and execution. The interface takes the form of a library of Ada package specifications and enables flexible, open-ended addition of new APSE tools.
- (3) Access control to APSE toolset is built-in.

The APSE toolset is a collection of programs that are written completely in Ada and make use of the above mentioned KAPSE interface. Therefore all these tools are host independent and fully portable. However, several tools are target dependent, for example, the code generator, parts of the linker, the loader, and the debugger. These must be re-written to support a new target. We can note however that the front-end of the Ada compiler is designed to generate intermediate code which is independent of target computer.

As all components will be written in Ada, portability will be greatly simplified.

c. Portability of the APSE Database

Information held in the APSE database may need to be transported from one host to another. A change of host is often forced when a system moves from its development phase to its operational phase.

Two methods are possible for transportation of information: a data link connection between hosts or by intermediate dumping of the data on magnetic media. In addition, an APSE program is required to provide the necessary conversion between the APSE database and input/output.

This kind of portability is a nontrivial problem. Even when the original host and the new host have compatible basic environments that allow tools and database objects to be transported, there can be problems in determining precisely what should be moved to the new host. There are dangers in moving too little information to the new host, but it is also undesirable to move too much information.

d. Portability of Trained Personnel

Programmer portability refers to the ability of programmers to move from one project to another without extensive training.

The US DoD's goal is to standardize the Ada language, admitting no subsets or dialects, and to standardize the majority of the APSE user interface. Therefore, programmers and other APSE users should have no difficulty using another APSE system.

5. The Use of Ada and Its Support Tools

In the design of the Ada language and of the APSE facilities are contained state of the art ideas for real-time software design and construction. The early notion of building real-time software as a sequence of actions initiated regularly by external events has given way to structuring for

maximum asynchronous parallelism decoupled as far as possible from rigid external timing.

Ada extends the constructs for parallelism by adding completeness in the areas of time outs, multi-choice events, and fault handling. All this means that to get the most out of software written in Ada, advantage must be taken of the language constructs and traditional real-time systems design must be re-thought.

The APSE also opens more rewarding techniques for software development. For example, when development in a host environment is stressed it is natural to put more emphasis on testing with a simulated environment. A simulated environment can be employed profitably by doing software prototyping at an early stage in the requirements analysis or system design, using the highly productive Ada/APSE products.

There are several ways in which the APSE components could be used:

- a. The full APSE running on a large main-frame host computer in a time-sharing mode. Representative target computers are connected for check-out purposes. This represents the ideal usage leading to maximum life cycle cost savings.
- b. The full APSE, with other language systems integrated into and making use of the APSE facilities. Integration would be a significant redevelopment for the other language systems.
- c. Some of the APSE components are incorporated into the existing operating systems environment. For example, editors, compilers, loaders and debuggers can be run within the existing system's framework. However, many benefits would be lost.

C. SOFTWARE ENGINEERING IMPLICATIONS

The goal intended for the Ada program by the US DoD is primarily the reduction of its own expenditure on military software. This is computed across the complete life cycle of all software projects and although it may be estimated now, it may take over a decade or more before the projections can be confirmed. At the same time, it is hoped that the quality of the software products will improve, especially in the areas of reliability and ease of maintenance.

The mechanism for achieving this software improvement is the eventual acceptance of a single high order language, Ada, for writing defense system software and the use of a single standard support environment for the language, the APSE.

1. Benefits Using Ada

The claims which are listed below for Ada, indicate the valid benefits and how Ada's use would achieve life cycle cost savings.

- a. The development of the Ada language and the procurement of the APSE components is a once only expenditure. This includes any necessary development iterations to the components and enhancements to provide availability on different hosts and targets. For full benefits all projects must use the standards.
- b. All projects have access to full software development and software management tools in an environment which is conducive to productive programming.
- c. APSE is designed to allow different projects, which may be in different phases of their life cycles, to share its facilities.
- d. APSE can be implemented to coexist with other host computer environments. Thus the host computer might be used more cost effectively.

- e. The standardization avoids excessive dependence of the procurement authority on particular hardware manufacturers and software suppliers.
- f. The standardization of ADA and APSE encourages mobility of trained staff and interchange of expertise. This is particularly important in a defense context, as military personnel are often posted for relatively short periods.
- g. Ada and APSE assist in the production of higher quality software and higher development productivity. A high quality product is essential in the military context. It may be the difference between surviving and not surviving.

2. Penalties Using Ada

Many objections have been raised against Ada and against language standardization as a principle. The following is a list of potential penalties.

- a. Standardization might restrict the use or even development of future advances in software technology.
- b. Ada is a large and complex language. As a result:
 - (1) The likelihood of making syntax errors is increased, since human information processing capabilities are limited.
 - (2) Standardization efforts are not simple. The smaller the language the more likely the success of promoting a standard version. Already several subsets of Ada are commercially available.
 - (3) Teaching is difficult. The successful teaching of a new language to potential users is vital for its acceptance. Training time will be increased in comparison to other languages.
 - (4) It is so large that it is difficult to use. Even on large mainframe computers the resources required for compiling Ada code are substantial.
- c. If standardization is not achieved, the Ada effort only increases the number of languages in use.

It is considered that the potential penalties are small in comparison to the potential benefits. However, time is required to develop mature software products. As with any new item of software, faults, inadequacies etc., must be expected.

D. ADA'S CONTRIBUTION TO SOFTWARE QUALITY AND DEVELOPMENT PRODUCTIVITY

The promotion of higher software quality and higher development productivity was an implicit design goal of Ada and APSE. The Ada language has been derived in accordance with the US DoD's technical requirements of the Steelman document. Steelman requires the language to satisfy the needs of embedded computer applications. It should be implementable, machine independent, and should avoid unnecessary complexity. It should promote the production of reliable, maintainable, and efficient programs.

The Stoneman requirements for the support environment repeat similar criteria for the APSE toolset and additionally emphasize the need of portability. An overall appraisal of how successfully Ada and APSE will meet these goals is accomplished as follows.

1. Software Quality

As an aid to assessing the overall quality of a software product, the notion of quality can be broken into quality attributes. These attributes are defined in Section 4 and they are:

Accessibility
Accountability
Accuracy
Augmentability
Communicativeness
Completeness
Conciseness
Consistency
Correctness
Device efficiency
Device independence
Efficiency
Human engineering
Maintainability
Modifiability
Portability
Reliability
Reusability
Self-containedness
Self-descriptiveness
Structuredness
Testability
Understandability
Usability

At its higher level the quality or general utility of a software package can be divided into three areas:

Usability: How well (easily, reliably, efficiently) it can be used.

Maintainability: How easy it will be to maintain (understand, modify, retest).

Portability: Will it still be usable if the environment is changed?

However these three high level quality attributes can be broken into more basic components. For example, maintainability is a high level attribute which includes the lower level attributes of modifiability, understandability, and testability. In this tree-like structure, if a program is maintainable it must necessarily be understandable, testable, and modifiable. This structure extends to another level of more primitive concepts, thus, if a program is understandable it is also necessarily structured, consistent, concise, and self-descriptive.

Considering the trade-offs of the quality attributes we can say that some are in conflict with the others. Therefore, the applied technology is making a compromise. The most serious compromise when using a high order language such as Ada usually concerns efficiency:

- a. The use of a modular structured HOL enhances understandability, testability, modifiability, and portability at the expense of efficiency.
- b. The use of highly accurate numerical algorithms or floating point instead of integer arithmetic, can enhance reliability at the expense of efficiency.
- c. The use of robustness features such as extensive error checking enhances reliability and human engineering at the expense of efficiency.

- d. The use of communicativeness features such as detailed error messages, enhances testability and human engineering at the expense of efficiency.

2. The Impact of Ada on Software Quality

This section is addressed in Figure 1 where Ada's technical features (Para. B.2) are cross-referenced against the low level quality attributes. From this an assessment is made whether the feature promotes or hinders the attribute, e.g., use of the subtype constraints language feature promotes correctness in comparison with a technology without such a feature.

Overall, the picture indicates that the attributes portability, reliability, and maintainability are promoted.

3. Development Productivity

The use of Ada and APSE will have potential benefits in the following development activities:

a. Programming

It has been demonstrated that programmer productivity can be improved clearly through the use of a good high order language with good support facilities. Using Ada, better software will be produced more quickly, principally because of the increased human engineering.

b. Error Detection and Testing

The presence of high order constructs and data type compatibility checking means that the compiler can trap a large proportion of errors before any code is generated. The exception feature is available to handle unexpected

errors and modularity and structuring provided by the language are aids to efficient testing. Thus Ada and APSE promote earlier detection of programming errors via the compiler and test toolset.

c. Maintenance and Enhancement

These activities will be more efficient because of the improved design visibility and understandability of software. Maintenance for the correction of programming errors will be reduced because of the higher software quality and earlier error detection provided by Ada. An improvement in this area is of extreme importance because maintenance traditionally consumes a high proportion of the total life cycle expenditure.

d. Training

Ada standardization promises the possibility of transferring software and personnel between projects without large overheads.

4. Definitions of Software Quality Attributes

Accessibility

The code has the characteristic that it facilitates selective use of its parts (e.g., variable dimensioned arrays). Accessibility is necessary for efficiency, testability, and human engineering.

Accountability

The code has the characteristic that its usage can be measured. This means that critical segments of code can

be implemented with examinations to measure timing, whether specific branches are exercised, etc. Code used for examinations is preferably invoked by conditional inclusion techniques to eliminate the additional instruction words or added execution times when the measurements are not needed.

Accuracy

The code has the characteristic that its outputs are sufficiently precise to satisfy their intended use. This is a necessary attribute for reliability.

Augmentability

The code has the characteristic that it can easily accommodate expansion in component computational functions or data storage requirements. This is a necessary attribute for modifiability.

Communicativeness

The code has the characteristic that it facilitates the specification of inputs and provides outputs whose form and content are easy to understand and are useful. Communicativeness is necessary for testability and human engineering.

Completeness

The code has the characteristics that all its parts are present and each part is fully developed.

Conciseness

The code has the characteristic that excessive information is not present. This implies that programs are not excessively fragmented into modules, overlays,

functions and subroutines, nor that the same sequence of code is repeated in numerous places, rather than defining a subroutine or macro.

Consistency

(a) Internal consistency: The code has the characteristic that it contains uniform notation, and terminology within itself. Internal consistency implies that coding standards are homogeneously adhered to; e.g., comments should not be unnecessarily extensive at one place, and insufficiently informative at another.

(b) External consistency: The code has the characteristic that the content is traceable to the requirements. External consistency implies that there is a one to one relationship between functional flowchart entities and coded routines or modules.

Correctness

The code has the characteristic that there is agreement between the programs total response and the stated response in the functional requirements.

Device efficiency

The code has the characteristic that the operations, functions, or instructions provided by the code are performed without waste of resources with respect to the device. A program may be device efficient with respect to one device but not another, implying that it is not efficient with respect to the overall set of resources it employs.

Device independence

The code has the characteristic that it can be executed on computer hardware configurations other than its current one. This attribute is necessary for portability.

Efficiency

The code has the characteristic that it fulfills its purpose without waste of resources. This implies that choices of source code constructions are made in order to produce the minimum number of words of object code, or where alternate algorithms are available those taking the least time are chosen etc.

Human engineering

The code has the characteristic that it fulfills its purpose without wasting the users' time and energy. This attribute implies accessibility, robustness, and communicativeness.

Maintainability

The code has the characteristic that it facilitates updating to satisfy new requirements or to correct deficiencies. This implies that the code is understandable, testable, and modifiable.

Modifiability

The code has the characteristic that it facilitates the incorporation of changes once the nature of the desired change has been determined.

Portability

The code has the characteristic that it can be transferred to and operated easily and well on computer configurations other than its current one.

Reliability

The code has the characteristic that it can be expected to perform its intended functions satisfactorily. This implies that the program will compile, load, and execute, producing answers of the requisite accuracy. It also implies that it is complete and externally consistent.

Reusability

The code has the characteristic that it can continue to perform despite some violation of the assumptions in its specification. This implies, for example, that the program will properly handle inputs out of range.

Self-containedness

The code has the characteristic that it performs all its explicit and implicit functions within itself. Examples of implicit functions are initialization, input checking, diagnostics, etc.

Self-descriptiveness

The code has the characteristic that it contains enough information for a reader to determine or verify its objectives, assumptions, constraints, inputs/outputs, components. Self-descriptiveness is necessary for both testability and understandability.

Structuredness

The code has the characteristic that it possesses a definite pattern of organization of its interdependent parts. This implies that evolution of the program design has proceeded in an orderly and systematic manner, and that standard control structures have been followed in coding the program.

Testability

The code has the characteristic that it facilitates the establishment of verification criteria and supports evaluation of its performance. This implies that requirements are matched to specific modules or diagnostic capabilities are provided, etc.

Understandability

The code has the characteristic that its purpose is clear to the reader. This implies that variable names or symbols are used consistently, modules of code are self-descriptive, and the control structure is simple or in accordance with a prescribed standard.

Usability

The code has the characteristic that it is reliable, efficient, and human engineering has been applied. This implies that the function performed fulfills the requirements and the program is robust against human errors.

III. THE HELLENIC COMMAND CONTROL AND INFORMATION SYSTEM (HCCIS)

The purpose of this chapter is to provide an overview of the characteristics and requirements for HCCIS.

A. DEFINITION OF HCCIS

In peace time the General Staff's main function is to prepare and coordinate defense plans for all the Hellenic area. In war time it controls all the operations at land, sea, and air of this area.

The Hellenic Command Control and Information System (HCCIS) is a future software system which is of interest to Hellenic Forces and is intended to provide computer support to the command and control processes between several Commands. It uses automatic data processing equipment to collect, process, display and distribute the necessary information.

This system consists of personnel, procedures, facilities and equipment used to provide information and is composed of both automated and manual information systems. The system provides the capability to plan, direct, and control the forces during periods of peace and war. HCCIS supports all missions that are assigned to Hellenic Forces under all operating conditions. Designed to be modular equipment, the system has the flexibility to support combat operations regardless of the size and composition of the combat units.

HCCIS supports units in all operating environments including training exercises.

The full operational capability for the HCCIS is not expected before the mid-1990's. With potential software implementation on such a large scale project and with a production time-frame from 1988 through 1995, the Ada language system is clearly of relevance. Therefore we can establish the likely software engineering requirements of HCCIS and examine whether Ada/APSE features can meet these requirements.

B. COMMAND SUPPORT CONCEPT

This concept emphasizes individual performance at the Command level. Every Command would be equipped with a subsystem which supports the operational requirements of the Command.

Within each subsystem, the components of the subsystem would be based on automatic data processing functions to simplify design and maximize performance. These components would be interconnected using a high speed local area network.

The network configuration provides standard interfaces applicable for connecting automatic data processing elements. It also provides network access and network control. The network control is distributed among all the equipments connected to the network. It consists of monitoring the exchange of data that takes place between the ADP elements and also in managing the dynamic allocation and reallocation of

system resources. The network is an open system architecture. The automatic data processing resources at each Command are allowed to grow by the addition of new hardware modules as necessary. Major attributes of the network are:

Flexible topology to interconnect the automatic data processing resources for each of adding or removing equipment.

High reliability achieved with equipment replication.

All the Commands would be connected into the HCCIS data transfer service system. System management and control might be by HCCIS. However each Command manages and operates the equipment within its own facilities. The HCCIS system has an integrated data base for all appropriate operational requirement areas, and each Command makes use of this data base. Also each Command uses the standard data base management system which provides effective exchange of information between Commands and satisfies interoperability requirements. HCCIS provides the Commands with reports from land, sea, and air elements in contact with the enemy and serves as a channel for intelligence requests from these units. It incorporates automation aids to facilitate the information processing tasks. Specifically, it provides tools to simplify the following tasks:

Entry and storage of information.

Management, retrieval, and display of text and graphics.

Generation, analysis, and modification of plans.

Printing and dissemination of plans, orders and reports.

These tasks support the following major activities of each Command.

1. Receipt of Mission

The arrival of the initiating directive starts the preliminary planning process. The initiating directive for the assigned mission is entered into the HCCIS data base automatically or manually, depending on which form the message is received.

2. Commander's Analysis and Mission

The Commander's analysis of the initiating directive and the assigned mission is aided by the information made available to him by the information retrieval capabilities of HCCIS. Retrieval of previously stored information and intelligence allows the Commander to update his knowledge about the objective area.

3. Collection of Information

HCCIS provides the capability so that each planner has the best information available throughout the planning phase of the operation. Query messages can be generated and dispatched to obtain required information. The system aids message development by presenting the appropriate menu and by interacting with the operator.

4. Preparation of Planning Guidance

By considering what information is available, the Commander can enter his initial planning guidance into the data base and distribute it, using HCCIS. As more information

becomes available, the Commander's guidance can be easily modified with HCCIS text and graphic processing.

5. Development of Courses of Actions

In the development, analysis, and comparison of alternate courses of action, the planning staff will use the information management capabilities of HCCIS to ease the planning effort.

6. Development of Staff Estimates

The staff planners use the HCCIS information processing aids to enter, store, modify and disseminate the text and graphics involved to include the following:

Information processing.

Inter-system query.

Word processing.

Message generation.

Algorithmic calculations.

The staff prepares and enters into HCCIS a graphic presentation, supported by text, of the enemy's probable reaction to each course of action. These graphic and text depictions are stored into the data base for use during the combat and in briefing the Commander.

7. Briefing the Commander

The information presentation capabilities of HCCIS help the Commander to conceptualize the situation and weigh the relative merits of each course of action.

8. Commander's Decision

The Commander's ability to review, modify, and reevaluate the staff's information and recommendations aids in his decision process.

9. Preparation of the Concept of Operations

Once the Commander's decision has been made, the staff may finalize the selected course of action and the approved version of the concept of operations is automatically transmitted to other Commands served by HCCIS. Once the concept of operations is issued, the system provides automated assistance during the conduct of detailed planning.

C. HARDWARE CONFIGURATION

The structuring of each automatic data processing center in HCCIS can be seen as a cluster of hardware equipments such as computers and peripherals (discs, tapes, printers). Although, within the HCCIS the different processing functions can be supported by dedicated processors of different types, the system calls for commonality of equipment to such extent as it is operationally and technically feasible.

Users and operators use the automatic data processing system through work stations. Each work station consists of consoles configured from specific hardware elements such as interactive text display, interactive graphic display, monitor display, text printer and graphic printer. At the display station, which is the central part of each work station, an operator interacts with the system, reviews the

tactical situation, obtains real-time data, makes and implements decisions and receives and originates digital messages. The major equipment items which are appropriate to satisfy the HCCIS system requirements are the microcomputer, the dynamic display terminal, the large screen display, the printer, and the communication control panel.

1. Microcomputers

Microcomputers are used in all HCCIS centers to perform functions such as:

- a. Execute HCCIS application programs.
- b. Contain and maintain the database.
- c. Drive peripherals such as printers.
- d. Perform computations.
- e. Route messages.
- f. Support on-line maintenance.
- g. Provide digital communications access processing.

Each microcomputer can be programmed to perform a specific set of functions using prepared programs that are loaded into the machine.

2. Dynamic Display Terminals and Large Screen Display

On the display screen the operator is able to have both graphic and text display simultaneously. The selection of a display element, either graphic display or text display, by the operator initializes a search of the data base to retrieve all display elements of the type selected within the geographical area or whatever is being displayed on the

terminal. Once these elements are displayed, the operator may selectively erase portions of the display not required. Since the desired collection of elements is being displayed, the operator may assign an identifying name to the display. This name may then be used to regenerate the display by a single operator action at a later time, and the regenerated display shows current data. Graphic situation displays depict friendly and enemy support information and control measures in any combination desired on a real time basis.

High level Command functions such as the maintenance of situation assessment are supported on a large screen display.

3. Printers

Two types of printers are used in HCCIS centers: the small printer, and the large printer capable of printing text and graphic hard copy output.

4. Communications Control Panel

The communications control panel is used by the operator to interface with the voice communications network. The network consists of radios, switched voice circuits, and dedicated voice circuits. The panel is capable of monitoring access to any one or combination of radio nets and switched or dedicated voice circuits, and transmission access to any net or any combination of switched or dedicated voice circuits.

In addition Commands may have dedicated hardware support to guarantee availability and responsiveness for

time-critical applications. All essential hardware equipment is duplicated. The overall automatic data processing system configuration is controlled locally by an operational management processor.

D. SOFTWARE CONFIGURATION

System software is required for the application, data base, man/machine interface, communications, continuity of operations, security, interface unit and other functions. System software for process monitoring, buffer management, and database management is standardized to the maximum possible extent. Each of the processors connected to the network has system software to manage the data exchange between automatic data processing resources. The system software of the operational management processors performs control of the local resources. The application software packages will be designed to satisfy requirements of the Commands. The automatic data processing requirements of HCCIS can be satisfied from the following categories of application software.

1. Communications

HCCIS is a distributed data processing system and requires dependable communications. The primary means of communication between Commands is a switched multichannel system which handles high speed digital data message exchange. Using digital communications efficiently, the requirements for voice radio communications are reduced. However, voice communication between Commanders is still critical. The

greatest volume of the HCCIS communications requirements is digital data message traffic, both operator and machine generated, through common user message switches. The messages range from short real-time messages to longer narrative type messages. Voice communication is for rapid, person to person exchange to modify, amplify or supplement graphics and text information. Dedicated voice channels (hot lines) are required between HCCIS centers to provide rapid communications between specific key personnel.

The access between the HCCIS microcomputer and the communications network is provided by the communications access processing. The communication access processing performs all those functions which are required to interface with the various communications units while maintaining a standard interface to the microcomputer. It performs the following functions:

- Protocol execution and conversion.
- Signal format conversion.
- Data rate conversion and buffering.
- Message processing.
- Communications system control.

The actual formatting of messages is a combined function of the communications access processing and HCCIS operating system. The HCCIS routine which generates the message requirements provide the communications access process with precedence and classification. The communication access

processing is a combination of program and equipment functions. The equipment functions are associated with communications interface and performing routine tasks which include timing, forward error correction, buffering, control and monitor signaling. It is important to note that all HCCIS communications are secure.

2. Data Distribution

a. Data Base

A data base is a set of data stored in some special way in direct-access computer storage. This is the working set of data which the HCCIS system is responsible for maintaining. Data received by the system is stored in the data base and is then available concurrently to all the HCCIS centers. The identified data categories are:

- (1) Own capabilities.
- (2) Own situation.
- (3) Enemy capabilities.
- (4) Enemy situation.
- (5) Other forces.
- (6) Geographical environment.
- (7) Plans and rules.
- (8) Targets.
- (9) Weather.
- (10) Miscellaneous.

When a user in any center is waiting for the data or must act upon the data received, he is alerted that the data he needs is in the data base. In this way, every one

operates on the same and most current data and the transmission of messages, reports, plans, and other information to each center is reduced.

Everyone who operates in HCCIS views the data base as being entirely on the local computer and containing only the data needed for his functions. This is the logical view of the data. At each Command a scratch pad area is available for temporary, preliminary or unreviewed data which can only be accessed at the center that stores it. Once the data is finalized it is moved to the main data base, which then makes it available to the other centers. The HCCIS data base is distributed over all the HCCIS Commands. Although each user assumes that there is only one copy of each piece of data, there are multiple copies of some data. This redundant data is maintained by the database management system to improve processing speed and for data base recovery. Data no longer in use must be eliminated from the data base in order to control its size.

b. Database Management System

The database management system is the software that handles the storage and retrieval of the records in the data base. The DBMS cannot exist without a data base. The DBMS is the active partner and the data base is the passive one. As such the DBMS moves the data among the various computers so that it appears to the user that the entire data base is accessible to him. The HCCIS data base is accessed and controlled by the DBMS which performs the following functions.

(1) Provides Data Independence. This is the independence of the application programs to structural changes in the data base. If the programs that act on the data base do not need to be modified and possibly not even recompiled after the changes to the data base then it can be said that there is a degree of independence. This capability serves to minimize the reprocessing problem. Also the data independence allows users to have logical views of the data which differ from the physical structure.

(2) Access Controls. The DBMS provides the capability to protect sections of the data base from unauthorized access or modification. While the need for this kind of facility varies, it always becomes more critical if the data base is to be accessed from on-line terminals. These controls protect classified data from access by un-cleared users.

(3) The Data Base Access. The DBMS performs the data base accesses for the HCCIS application programs.

(4) Data Integrity. This is the currency, accuracy, and readability of the data stored in the data base. Integrity is maintained by a combination of DBMS functions, equipment, and procedures within the Command.

(5) Data Base Recovery. Recovery is the restoration of the data base after a failure, to a state that is acceptable to the users. What is "acceptable" may vary from situation to situation, but it means at least a consistent state with no erroneous data.

c. Query Language

The data base query language operates in conjunction with the DBMS and provides the HCCIS user with the ability to:

- (1) Perform queries to the HCCIS data base.
- (2) Direct the data base query language to store query messages for future and repeated use.
- (3) Direct the data base query language to send output to any of the available output devices.

This query language can give for example the location of each unit within a geographical area and its status.

3. Messages and Reports

Since a message or a report format has been established and entered into the data base, it can be recalled and used by the operator. The format contains standard information which is known to the system, such as date/time/group, standard headers and routine indicators. Dissemination of messages and reports is aided by tables of distribution which need only to be entered into the HCCIS data base once, with subsequent modifications to reflect changes.

4. Information Processing Tools

Information processing tools are the following.

a. Text Entry

The entry of a text into the HCCIS is achieved via the keyboard of the terminal unless the text is in a machine

readable form. Formatting is controlled by page size and character counts for line and page totals.

b. Text Editing

This is needed for corrections and editing of material using the text editing functions insert, delete, replace and print.

c. Workload Administrator

This is a combination program and operator procedure which allows operators to process more than one task at a time according to their set of priorities. This is accomplished by the HCCIS system's maintaining two queues, an incoming message queue and an action queue.

d. Task Identification

The two workload administrator queues are lists of tasks which are awaiting operator action. Each entry in the queue is identified by an abstract of the task which allows operator recognition.

e. Task Queueing

The workload administrator will order tasks in the queues according to priorities. When the priority of the message is unspecified, the system will execute a default routine.

5. Tactical Processing Tools

Tactical processing tools are the following.

a. Algorithmic Calculations

HCCIS provides a variety of automated calculations, available upon request, to assist the Commander and his

staff. The rapidity and accuracy of these calculations facilitates the investigation of alternative courses of action.

b. Tactical Simulation

Within HCCIS, the capability to simulate proposed tactics provides detailed measurement of the effectiveness of the elements of the plan. Parts of the simulation are: unit movement, snapshot simulation display, casualty damage assessment.

6. Continuity of Operations

In order to accomplish his mission, a Commander requires an uninterrupted flow of timely and accurate tactical information upon which to make tactical decisions. Thus, each Commander needs continuity of the flow of information for command, HCCIS operations, and combat operation. A serious interruption in the flow of information affects the continuity of command and the continuity of combat operations. This continuity is based on the following critical functions:

Transmit and receive digital messages.

Display friendly locations.

Display control measures.

Maintain data base.

HCCIS support ranges from a full system operation to complete failure of the system so that no system support is possible. The system degradation is characterized as minor or major. In minor degradation an HCCIS center is experiencing equipment or program malfunction or failure but HCCIS support

of the Command is not significantly impaired. In major degradation an HCCIS center is experiencing equipment or program malfunction or failure to a degree that significantly influences the ability of HCCIS to support the Command in carrying out its mission. Means and techniques used to alleviate the effects of degraded HCCIS operations are the following.

a. Spare Equipment

Degraded system performance is detected by operator observation and by programs that continuously monitor system performance (operational management processor). When system degradation is detected, the operational management processor or the operator executes programs that isolate the failure to a particular item of the equipment. The operator replaces the failed equipment with a spare (a quantity of spare equipments is stored) and restores the center to full operation.

b. Use of Voice Radio

Voice radio nets are maintained, at all times, regardless of the operating status of HCCIS. Thus with increased use of voice radio we can reduce the influence of system degradation.

c. Use Capabilities of Alternate Center

A degraded center may use the capabilities of a fully operable center to perform processing functions. Input data may be provided by the degraded center. Upon request of the degraded center, the operable center searches its

files for the information, compiles the data and processing results are sent to the requesting center. When system degradation affects a center's capability to receive digital messages, the storage capability of an operable center may be used to store messages until the degraded center is restored.

d. Hard Copy Printout

Periodically, a hard copy of the friendly and enemy situation display is produced automatically or on request of an operator. Hard copy printouts of information which are considered critical for the combat operations are also periodically produced by HCCIS. When a center's display capability has completely failed, the most current hard copy printouts are used to construct a situation display.

E. PERSONNEL

Although Command functions and structure are not altered with the introduction of HCCIS into their combat centers, procedures and methods differ considerably from current manual means. These changes lead to requirements for training to develop skills and knowledge not previously necessary within a center, such as interaction with input keyboards. No additional maintenance personnel is required at the organizational level, since the only maintenance to be performed to repair the failed equipments, is replacement.

Exercises provide the opportunity to establish demanding requirements of large scale tactical environments. During exercises operators demonstrate their ability in solving

problems related to maintenance and system operation. HCCIS has the capability of battle simulation, to generate simulated inputs and to provide realistic feedback for operator actions in order to improve the proficiency of HCCIS users. Some other on-the-job training capabilities which aid personnel in the development and maintenance skills, are:

Operator interaction.

System diagnostic text messages.

System performance monitoring.

Fault isolation.

F. RELIABILITY AND MAINTAINABILITY

1. Reliability

Reliability can be formulated as the probability that the system will perform its intended function over the stated duration of time in the specified environment for its usage. An HCCIS center is in failed condition when it is not capable to support minimum essential Command functions. Equipment reliability requirements can be stated in terms of mean time between failure. Center reliability can be stated in terms of the probability of not reaching a failure condition within a specified period. A minimum mean time between failures for each HCCIS center has to be determined which reflects the mean time between equipment failure occurring at a particular center.

2. Maintainability

Maintainability is a characteristic of design and installation which is expressed as the probability that an item of equipment, center or system will be restored to specified conditions within a given period of time when maintenance action is performed in accordance with prescribed procedures and resources. Center maintainability is defined in terms of the time required to restore a center to its specified operating condition. This time includes the time required to identify, remove and replace the failed item and also includes the time required to manipulate the software portion of the system to achieve full operability.

Also the system must be available all the time and individual centers must be available at a very high percentage of the time.

G. SECURITY

The protection of HCCIS is vital and imposes stringent requirements for system security. The primary HCCIS elements which implement this protection are HCCIS equipment and computer program features. These features provide mutually supporting protection of classified material as well as the integrity of the system itself. Security measures are the following.

1. Physical Protection

Physical protection measures are intended to reduce or prevent disruptions to operations or loss of assets.

These include access key locks, visual and audible alarms, and sensors to identify interruption in device operation.

2. Error Detection

It is necessary for HCCIS equipments to have the ability of detecting errors to minimize the probability of inadvertent disclosure. Error detection functions monitor memory and register operation whenever possible, causing automatic interrupts to security programming.

3. Operational Features

HCCIS processors have security protection features (e.g., interrupt system) designed into them in order to accomplish security objectives.

4. Access Control

Disclosure or dissemination of classified material or tactical information from unauthorized sources is absolutely prohibited. Thus, the HCCIS system is required to prevent this undesired situation. Some measures are: user identification, authentication, security system monitoring, and terminal operator training.

H. SOFTWARE ENGINEERING REQUIREMENTS

The implementation of the HCCIS with the characteristics identified in the previous section would require software engineering tools which give effective support for:

1. The programming of each type of computer to be used (mainframe, mini, micro).
2. Large programs development.

3. "Off-the-shelf" software packages (e.g., database management system, perhaps modified to suit HCCIS peculiarities whose interfaces are independent of the hardware used, and are compatible with the programming language used for applications. These would include:
 - (a) Operating system and device drivers.
 - (b) Database management system, including alerters and query language.
 - (c) Networking (inter-command) including protocol handling.
 - (d) Local network protocol handling at low and high levels.
4. High quality software product especially in the areas of reliability, maintainability and portability.
5. Real-time processing.
6. Security of data.
7. Concurrent execution of the same program but with different data sets.
8. Structured programming of software modules, authentication, security system monitoring, terminal operator training.

IV. APPLICATION OF ADA/APSE TO THE HCCIS

The previous chapters describe two assessments. Firstly, the products of the US DoD's Ada program were examined from a software engineering and software management viewpoint and the potential facilities, benefits and penalties identified. Secondly, the software engineering and management requirements for the implementation of HCCIS have been identified to the extent possible at this time. The rationale of this chapter is to compare the two for compatibility. Specifically, the comparison is made under the following headings:

Technically, are the software requirements of the target HCCIS met by Ada/APSE?

Will Ada be cost effective in comparison to alternative software technologies?

A. TECHNICAL COMPARISON

The software engineering requirements identified in Section D of Chapter III are compared here with the facilities to be provided by Ada/APSE.

1. The Programming of Each Computer Type to be Used

A major effort of the Ada system is to ensure that software is portable among the number of different hardware architectures on which the programs written in the language must execute. It is also necessary to ensure that an Ada program will produce the same results independent of which compiler translates it and the computer on which it is executed. Support of such requirements was a fundamental

design goal of APSE. APSE provides a target-independent program development environment on a central host computer system.

Since HCCIS requirements can be satisfied using microcomputers we can say that among the current technologies for microcomputers, the Ada programming language is the best. Nettleton [Ref. 14], in his article on Embedded Microcomputers, comparing several languages, says:

On a scale of 1 to 10, CORAL comes about 5 with Pascal at 6, FORTRAN at 4, assembler at 1. So what comes to 10? Ada, the new programming language for the US DoD, is said by many to be the ideal language for embedded computer applications. It is based on Pascal but has a lot of extra features for embedded applications and unlike many other languages, is rigorously defined. Ada will also be well supported and will be available for all commonly used computers.

Also in the article, "Ada for the Intel 432 Microcomputer" by Zeigler et al. [Ref. 13], we note, "Ada represents a new era in language standards and software portability. It is the primary language of Intel's new micromainframe, which directly supports many of its features".

2. Large Program Development

HCCIS is designed to support the training of Hellenic Forces by giving to the Supreme Commander all the facilities to plan, direct and control the Forces during periods of peace and war. This implies that the development of HCCIS consists of a large scale programming effort which requires a programming language with many features to support it.

Ada and APSE are designed with the fundamental design goal to support the development of large programs that will

be used over a long period of time. This can be achieved principally through the separate compilation and library facilities. Another feature that encourages programming in the large is the package feature. Of course, APSE is designed to support large teams of programmers as mentioned in Section B.3.a of Chapter II.

3. "Off-the-Shelf" Hardware-Independent Software Packages

a. Operating system and peripheral drivers. The Ada run-time support library provides the equivalent of the traditional real-time executive plus drivers for the standard peripherals (Section B.2.g(1) of Chapter II). Drivers for any new standard peripherals must be procured.

b. Packages for DBMS and networking. It is quite possible that application-oriented packages will be developed for the Ada target systems over the next few years. For example, a DBMS ("Adaplex") is currently being developed by Computer Corporation of America. However it is not certain whether any such development will meet the particular HCCIS requirements. If not, they might need to be procured, either written totally in Ada or with an Ada compatible interface which can be standardized across HCCIS.

4. High Quality Software

As discussed (Section C.1.g of Chapter II), the use of Ada and APSE has a major contribution in the promotion of software quality and development productivity, particularly in the areas of reliability, maintainability and portability. While Ada incorporates most of the features found in

programming languages such as FORTRAN, COBOL, PL/1 and Pascal, it also provides a set of features that support well-accepted software engineering principles. Ada produces software that is more:

(a) Reliable. Strong typing and procedures and module specifications reinforce consistent and complete coding; most of the software written in Ada can be checked and verified at compile-time; and the high-level nature of Ada reduces the amount of coding and, therefore, the chances for error.

(b) Portable. As a standard language, Ada must be supported in the same manner on many different systems. Ada is highly machine independent which supports more generalized solutions. Machine dependencies can be isolated and introduced in a very controlled manner through the use of packages and representation specifications.

(c) Readable and Maintainable. Abstract data types, especially enumerated types, enable the program to be defined in problem-oriented terms and values which enhance documentation. Language constructs and terms are employed in a well-defined consistent manner and are very English-like, which also enhances documentation. The modular structure of Ada (i.e., subprograms, modules, and separate compilation) supports the loose coupling of software which, in turn, improves maintainability.

All these features have a significant contribution in the promotion of software quality.

5. Real-Time Processing

An obvious objective on which HCCIS system is based is real-time processing, because it is important and necessary for most military programming actions.

The Ada language is the best for this purpose. No other language can improve on Ada in this area. This language provides explicit constructs for parallel processing such as mutual exclusion, real time interrupts, asynchronous termination. Ada contains the fundamentals of a real time executive. Presently such executives are implemented via several routines particular to each operating system. In Ada, desired executive control and synchronization of independent tasks can be obtained by proper selection of built-in language constructs. Incorporation of these features directly in the language not only reduces implementation efforts but also establishes a consistent approach across systems. Also the Ada language provides constructs for the real time interfacing of external devices to their software device drivers (Section B.2.c of Chapter II). The Ada library holds the routines for run time support, which must be procured for each new target computer.

6. Security of Data

In the case of military data bases, the information is considered to be of such value that no cost is spared to insure data security. Therefore security of data is extremely important for HCCIS system.

The database management system is responsible to provide this security. This is the appropriate software function to protect data from unauthorized access. At this time only very specialized hardware and operating system's features permit secure systems to be built. Currently no validated secure systems exist, however Ada Plex DBMS is now under development and it is believed that after a few years it will be available in the form which has solved the security problem.

7. Concurrent Execution of the Same Program

This very useful facility for the HCCIS system can be achieved by using re-entrant code. Multiple users can execute the same program in the main memory without needing their own copies of code. Production of re-entrant code is more complicated, but well worth the effort because of substantial savings in the use of memory. The Ada language supports the production of re-entrant code.

8. Structured Programming

Ada is 100% suited in this area. It is a strongly typed language in the sense that the type of every variable and expression can be determined at compile time and checked by the compiler. This reduces the run-time errors in programs and enhances reliability as mentioned above. Program modules include subprograms, packages and tasks in a block structured format which enables nesting. The control structures (conditional branching, case statement, iteration structure and exit statement) are well engineered to support structured coding.

In summary the combination of Ada and APSE is judged to fulfill the technical requirements for use in HCCIS. Some special packages (e.g., DBMS) may need to be procured to suit the special real-time requirements of HCCIS. Also, the implementation of secure systems which have users at various levels of security is a challenge which is independent of Ada. However, the use of Ada will probably make the implementation easier.

B. ALTERNATIVES TO ADA

There are several programming languages which may be used in HCCIS and the differences between them can have a significant effect on the quality of the finished product. The languages which are examined are the present military programming languages: Assembly, FORTRAN 77, CMS-2, JOVIAL J73, and the new one, Ada.

1. Assembly

Assembler code is always a first choice for many because it is immediately understandable if you know the computer's hardware and, for a small program at least, requires little software expertise. However, assembler code is tedious to write and difficult to read. This code is fine for small programs, but coding a large system in assembler code is not economical at all.

2. FORTRAN

Fortran is dominated by its orientation to scientific, numeric computing, but it provides low level control structures,

I/O functions, libraries, abstraction facilities and weak typing. Fortran has no real-time processing features or exception handling and leads to low productivity.

3. CMS-2

Although CMS-2 corrects some shortcomings found in Fortran, it still suffers in strong typing, and the 150 keywords indicate its complexity for both implementation and maintenance programmers. Also, and most important, CMS-2 has no real-time constructs and its reliability is limited.

4. JOVIAL J73

Beyond CMS-2, J73 has included the basis for strong typing, fundamental exception handling, tighter control of functions and procedures and slight improvements in control structures. The overall effect of these features is an increase in reliability and maintainability. J73 introduces several improvements to functions and procedures but it has no real-time structures and its portability is limited.

When considering other languages for use in HCCIS the following points are relevant.

The Ada language defines constructs sufficient for complete programming of real-time systems. The MAPSE provides the minimum of facilities necessary for efficient software development. The full APSE provides further highly desirable facilities.

The full cost benefits of using Ada depend on its standardization. In cases where a single language can be used

exclusively, Ada and APSE are likely to be the most cost effective technology.

V. CONCLUSIONS

This thesis has examined the mutual relevance of two programs: The US DoD's Ada high order language standardization program, and the Hellenic Command Control and Information System program.

The US DoD's Ada high order language standardization program is approximately at the half way stage, with the language definition complete but with the requirements for the Ada Programming Support Environment still being refined. The program's outcome can be predicted with some confidence.

The Hellenic Command Control and Information System is a future program expected to start in 1988.

The following points summarize the conclusion of this thesis.

- a. Ada will be the first language system to provide constructs for the complete programming of embedded computer systems, together with full programming support.
- b. Ada's existence is assured through the continuing backing of the US DoD. Experimental compilers are available now and an approved APSE is expected in 1984/1985. The US DoD intends all of its new embedded computer systems projects to be using Ada by the end of the decade.

c. Ada and APSE should offer substantial benefits to military software procurement, especially in the areas of software quality and productivity, while having minimal penalties. It is considered that Ada and APSE will be the best language system to be available in the mid-1980's.

This thesis examined the programming language requirements of HCCIS. These can be satisfied by Ada, and APSE could provide the necessary user support. Software systems programmed in Ada require no external operating system. However, the HCCIS program might need to procure software packages, not found in Ada/APSE, such as database management, network handling, and non-standard peripheral drivers. The timescales for HCCIS software development of 1988 to 1995 should be met by the availability of approved APSE products in 1984/1985. By the end of the 1980's, the APSE will be widely available and there will be much experience in its use.

LIST OF REFERENCES

1. Skees, W.D., Computer Software for Data Communications, 1981.
2. Jensen, R.W. and Tonies, C.C., Software Engineering, Prentice-Hall, 1979.
3. Grenn, T.F., Software Error Detection Model, M. S. Thesis, Naval Postgraduate School, Monterey, 1975.
4. Baker, C.E., The Navy's Automated Command Management Information System, M.S. Thesis, Naval Postgraduate School, Monterey, 1972.
5. Clifford, G.A., An Analysis of Requirements for the Development of an Intelligence and Communication Model, M.S. Thesis, Naval Postgraduate School, Monterey, 1979.
6. Johnston, F.W. and Dawson, M.T., An Appreciation of Army Divisional Command and Control, M.S. Thesis, Naval Postgraduate School, Monterey, 1980.
7. Rogers, M.A. and Myers, L.M., An Adaptation of the Ada Language for Machine Generated Compilers, M.S. Thesis, Naval Postgraduate School, Monterey, 1980.
8. Smith, S., "Requirements Definition and Design Guidelines for MMI," IEEE National Aerospace Electronics, V. 1, 1980.
9. Scheer, L.S. and McClimes, M.G., "DoD's Ada Compared to Present Military Standards HOLs. A Look at New Capabilities," IEEE National Aerospace Electronics, V. 1, 1980.
10. Ledgard, H., Ada--An Introduction, Springer-Verlag, 1980.
11. Barnes, J.G.P., "An Overview of Ada," Software--Practice and Experience, V. 10, November, 1980.
12. Wolfe, M.I., and others, "The Ada Language System," Computer, V. 14, June 1980.
13. Zeigler, S., and others, "Ada for the Intel 432 Micro-computer," Computer, V. 14, June 1981.
14. Nettleton, C.C.F., "Embedded Microcomputers," Mini-Micro Software, V. 6, 1981.
15. Carlson, W.E., "Ada: A Promising Beginning," Computer, V. 14, June 1981.

16. Brender, R.F. and Nassi, I.R., "What is Ada," Computer, V. 14, June 1981.
17. Wegner, P., "A Self-Assessment Procedure Dealing with the Programming Language Ada," Communications of ACM, V. 24, October 1981.
18. Shaw, M., and others, "A Comparison of Programming Languages for Software Engineering," Software--Practice and Experience, V. 11, 1981.
19. Shumate, K.C., "Ada--New Language that will Impact Commercial Users," Data Management, V. 19, August 1981.
20. Eventoff, W., Anderson, G., and Price, R., "Ada: A Significant Software Engineering Tool," Mini-Micro Systems, V. 14, April 1981.
21. Stenning, V., and others, "The Ada Environment: A Perspective," Computer, June 1981.
22. Goodenough, J., "The Ada Compiler Validation Capability," Computer, June 1981.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3. Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4. Professor Uno R. Kodres, Code 52Kr Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5. Professor Bruce J. MacLennan, Code 52M1 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
6. Hellenic Navy General Staff Branch A Stratopedon Papagou, Holargos Athens, GREECE	3
7. CDR Apostolos Koutsotolis Leoforos Papagou 69 Athens, GREECE	2

ATE
LME
8